УДК 519.72

Л.А. Пономаренко, В.А. Филатов

МОДЕЛЬ СИСТЕМЫ УПРАВЛЕНИЯ ИНФОРМАЦИОННЫМИ РЕСУРСАМИ НА ОСНОВЕ ТЕХНОЛОГИИ ПРОГРАММНЫХ АГЕНТОВ

Рассматриваются вопросы построения модели программного агента для автономного администрирования информационных ресурсов как абстрактного типа данных. Теоретически обоснован подход к построению такой модели на основании концепции алгебраических систем.

Введение в проблему

Информационные ресурсы современных вычислительных систем на базе сетей персональных компьютеров сосредоточены в файлах различных типов – от текстовых до файлов баз данных. Как правило, определенный тип файлов а, следовательно, и находящиеся в них данные обрабатывается своим типом программных средств. Так, например, файлы с расширением *.txt, *.rtf, *.doc находятся под управлением программных систем категории "текстовые редакторы" – WORD, WORDPAD, WORDPRO, файлы баз данных *.dbf, *.mdb – являются основой в программных комплексах получивших название "системы управления базами данных" (СУБД) и т.д. Такая специализированная информационная технология по мере ее внедрения и освоения начинает создавать пользователям некоторые неудобства при решении сложных, взаимосвязанных комплексов задач по технологии потоков работ "workflow" [1].

Под термином взаимосвязанный комплекс задач понимается последовательный или параллельный (может быть и комбинированный) набор законченных функциональных задач, решаемых различными программными системами. Например, в высшем учебном заведении с развитой информационной инфраструктурой независимо друг от друга решаются такие задачи:

- формирование и издание приказа о назначении стипендии (текстовый редактор WORD);
- выбор из баз данных всех деканатов студентов, успешно сдавших сессию (информационная система «Деканат» на основе СУБД);
- передача списка студентов в бухгалтерию и начисление стипендии студентам (ручная обработка списков и начисление стипендии в бухгалтерской системе на основе СУБД);
- формирование финансовых документов (платежных поручений) для бронирования и получения наличности (или перечисление денег на пластиковые карточки).

Рассмотренная выше информационная технология начисления стипендии стабильна, отработана годами и, как правило, функционально реализована в различных программных средах. Узким местом такого процесса являются этапы стыковки разнородной информации «выход»–«вход». Так приказ о начислении готовит и распространяет канцелярия, после получения приказа в деканате готовят списки студентов, далее передают эти списки в бухгалтерию и т.д.

В рассмотренном контексте может быть поставлена качественно новая задача интеграции информационных ресурсов: создание сценария выполнения

последовательности функциональных задач, согласование промежуточных форматов данных и автономное их выполнение.

Для решения поставленной задачи – выбора программной платформы, разработки модели программных агентов и управляемых процессов рассмотрим существующие программные средства и информационные технологии, получившие наибольшее распространение в настоящее время.

Обзор современных программных средств и технологий для решения задач автоматизированной обработки информации

Программные средства автоматизированной обработки информации можно разделить на следующие категории:

- системное программное обеспечение;
- промежуточное программное обеспечение;
- прикладное программное обеспечение.

Подробно рассмотрим промежуточное программное обеспечение (middlware), которое в настоящее время становится наиболее важной составляющей программных средств в технологии автоматизированной обработки информации. Оно выполняет функции прослойки между операционной системой и прикладной программой. Промежуточное ПО можно разделить на три группы:

- офисное,
- групповое,
- промышленное.

Это деление, конечно, условное, так как программное обеспечение каждого класса обладает некоторыми «общими» свойствами, позволяющими применить его в следующем классе (хотя и с некоторыми ограничениями).

Современное офисное обеспечение представляет собой программные пакеты, в состав которых входят текстовые процессоры, электронные таблицы, системы подготовки презентаций, персональные СУБД. Офисные пакеты интегрированы, что позволяет их составным частям взаимодействовать между собой и изготавливать композитные документы. Безусловным лидером на рынке офисных пакетов является Microsoft Office. В его состав входят программные средства, обеспечивающие все вышеназванные функции (Word, Excel, PowerPoint, Access), форматы файлов MS Office (прежде всего – doc) стали де-факто стандартом. Однако, лидерство MS Office имеет причиной не столько высокое качество продукта, сколько своевременное появление его на рынке и политику фирмы, создающей намеренные трудности для конкурентов в обеспечении совместимости их продуктов с MS Office. По-видимому, наиболее значительными из конкурирующих пакетов являются StarOffice (фирма Sun) и SmartSuit (фирма Lotus, подразделение IBM). Оба названных пакета, по крайней мере, не уступают MS Office в функциональности, превосходят его в надежности и к тому же являются многоплатформенными.

Для систем обработки данных, функционирующих в рамках офисного подхода одних средств, предоставляемых офисными пакетами, оказывается недостаточно. Современные организации представляют собой совокупность подразделений обменивающихся между собой информацией и выполняющих отдельные части общей работы. Основными фазами жизни информации в офисе являются:

- ввод информации в систему,
- хранение, навигация, поиск и фильтрация документов,
- коллективная работа с документами,
- вывод информации из системы.

Офисные пакеты применяются в рамках такого подхода для обеспечения первой и последней фазы, прочие же фазы поддерживаются программным обеспечением, строящимся на технологиях groupware и workflow [2].

Методика groupware ориентирована на небольшие рабочие группы для поддержки выполнения одной коллективной задачи при отсутствии дополнительной организационной структуризации, которая ограничивается обеспечением коллективного входа с помощью различных методов доступа:

- сетевой доступ к файлам и базе данных;
- локальная и глобальная электронная почта (включая конференции и дискуссии);
- терминальный доступ, пересылка файлов и электронная доска объявлений;
- просмотр и интерпретация гипертекста (гипермедиа).

Технологии workflow служат для автоматизации документооборота в средних и крупных офисах и для них характерны:

- поддержка многопользовательской работы с несколькими задачами одновременно;
- четкая структуризация выполнения работ по ролям и документам с контролем исполнения.

Во всех случаях в процессе коллективной работы большое значение имеют средства разрешения конфликтов при совместном использовании ресурсов, санкционирование входа по идентификаторам и паролям, защита информации с помощью администрирования прав доступа. Дополнительный уровень безопасности поддерживается методами и средствами шифрования и электронной подписи.

Среди программного обеспечения, обеспечивающего интегрированный подход к коллективной работе, лидером является многоплатформенная система Notes фирмы Lotus (подразделение IBM). Lotus Notes представляет собой платформу типа "клиент-сервер", служащую для разработки и размещения приложений класса groupware на основе электронной почты. Она позволяет пользователям получать, отслеживать, совместно использовать и создавать документальную информацию, получаемую из различных источников, таких как прикладные и оперативные системы, сканеры или факс-аппараты. Клиентам Notes предоставляет доступ к сети через любой применяемый ими пользовательский интерфейс (Windows, Mac, OS/2, Unix).

Notes обеспечивает:

- единый, постоянный пользовательский интерфейс для обращения ко всем абонентам, ресурсам и информации;
- гибкость при обработке сложных композитных документов;
- быструю разработку прикладного ПО для рабочих групп;
- систему защиты доступа к информации на всех уровнях;
- тиражирование информации, независимо от места ее расположения;
- открытость (поддержка разных сетевых ОС и приложений, внешних источников данных, систем электронной почты и API);
- масштабируемость (от рабочей группы из двух пользователей до корпоративной сети);
- интеграцию набора разнообразных элементов клиентских и серверных программных модулей.

Однако, Notes не владеет монополией на рынке группового программного обеспечения. Наряду с этой системой применяются и другие, обеспечивающие как интегрированное решение задач коллективной работы (например, Microsoft ExchangeServer), так и отдельных задач этого класса – прежде всего - управления электронными документами (например, DOCS OPEN компании PC DOCS Inc.,

Excalibur EFS фирмы Excalibur Technologies Corp.), автоматизации документооборота (например, Action Workflow компании Action Technologies).

Современные решения промежуточного ПО обеспечивают межпрограммное и межплатформенное взаимодействие и реализуют платформу для среднего звена – сервера приложений, обеспечивая обширный набор необходимых служб: управления транзакциями, защиты и т.д. Наряду с выполнением прикладных программ, промежуточное ПО играет также важную роль интегрирующего компонента, изолируя приложения от сетевых протоколов и деталей операционных систем.

Безусловным лидером на рынке промышленного промежуточного ПО является фирма IBM, продукты и интегрированные решения которой во всех подразделениях этого класса входят в число сильнейших и доступны на множестве аппаратно-программных платформ как IBM, так и других производителей. Практически полный спектр подобных продуктов производит фирма Microsoft, однако, ее продукты работают только в средах Windows. Наряду с “гигантами” на этом рынке выступают и фирмы, являющиеся традиционными поставщиками средств разработки приложений (напр., BEA Systems, Inprise, PowerSoft).

Для разработки объектно-ориентированных приложений потребовалась специальная система промежуточного ПО, способная интегрировать объекты на удаленных платформах. На сегодняшний день существуют два варианта спецификаций на объектное промежуточное ПО – стандартная архитектура брокеров объектных запросов CORBA, поддерживаемая консорциумом OMG, и разработка COM/DCOM компании Microsoft. Обе распространяют принципы вызова удаленных процедур на объектные распределенные приложения и обеспечивают прозрачность реализации и физического размещения серверного объекта для клиентской части приложения; поддерживают возможность взаимодействия объектов, созданных на различных объектно-ориентированных языках и скрывают от приложения детали сетевого взаимодействия. И DCOM, и CORBA, дают возможность динамического связывания удаленных объектов (клиент может обратиться к серверу-объекту во время выполнения, не имея информации об этом объекте на этапе компиляции). В CORBA для этого существует специальный интерфейс динамического вызова, а COM использует механизм OLE.

Информацию о доступных объектах сервера на этапе выполнения клиентская часть программы получает из специального хранилища метаданных об объектах – репозитария интерфейсов (CORBA) или библиотеки типов (DCOM). Эта возможность очень ценна для больших распределенных приложений, поскольку позволяет менять и расширять функциональность серверов, не внося существенных изменений в код клиентских компонентов. Существуют десятки реализаций CORBA: IBM ComponentBroker, Iona Orbix, Inprise VisiBroker, BEA ObjectBroker. Основная реализация COM принадлежит Microsoft и интегрирована в Windows. С помощью своих партнеров Microsoft предпринимает попытки экспортировать COM и на другие операционные системы. Однако, CORBA, изначально ориентированная на кроссплатформенную поддержку и реализованная для всех разновидностей Unix, всех версий Windows и многих других важных ОС, имеет очевидное преимущество перед объектной моделью Microsoft.

В настоящее время возникла тенденция интеграции промежуточного ПО в серверы приложений, выполняющих весь комплекс перечисленных выше функций для корпоративной системы и обеспечивающих прозрачное взаимодействие между ее разнородными узлами и с внешними системами (“любой клиент – любой сервер”). Появление серверов приложений как отдельных готовых решений связано и с бурным вторжением Internet в сферу корпоративных высоко-критичных систем.

Многоплатформенные Web-системы, как правило, делают ставку на CORBA и на модель построения серверных компонентов Enterprise JavaBeans (EJB). Хотя EJB имеет свой собственный механизм взаимодействия клиентских и серверных Java-объектов RMI (remote method invocation) EJB и CORBA сегодня рассматриваются как взаимодополняющие, а не как конкурирующие решения. Объекты CORBA и компоненты EJB инкапсулируют бизнес-логику. Для взаимодействия удаленных объектов могут использоваться механизмы как CORBA, так и EJB. EJB поддерживает механизм транзакций в соответствии со спецификацией JTS, которая, в свою очередь, базируется на CORBA Object Transaction Service. CORBA предоставляет ряд необходимых для сервера приложений служб: службу каталогов для поиска объектов в сети по имени, а не физическому адресу, службу оповещения о событиях, службу защиты и др. Наконец, EJB обеспечивает кроссплатформенность реализации, а брокеры объектных запросов позволяют интегрировать различные платформы [3].

Разработка концептуальной модели управления информационными ресурсами на основе технологии программных агентов

Определение мультиагентной системы управления информационными ресурсами вычислительной системы.

Предметная область. Предметной областью являются компоненты вычислительной среды: файлы, приложения, каталоги, диски, персональные компьютеры.

Класс систем. Система относится к классу надоперационых систем.

Функции системы. Манипулирование файлами, аудит информационных ресурсов, управление выполнением приложений по сценарию, интеграция распределенных гетерогенных баз данных и знаний

Функциональная структура. Программные агенты, система проектирования и управления мультиагентным пространством, интерфейс пользователя.

Средства представления информационных процессов.

Функционирование информационных систем носит целенаправленный характер. Типичным представителем такого функционирования является решение задачи планирования пути достижения нужной цели из некоторой фиксированной начальной ситуации. Результатом решения задачи должен быть план действий - частично-упорядоченная совокупность действий. Таким планом является сценарий, в котором в качестве отношения между вершинами выступают отношения типа: "цель-подцель" "цель-действие", "действие-результат" и т. п. Любой путь в этом сценарии, ведущий от вершины, соответствующей текущей ситуации, в любую из целевых вершин, определяет план действий.

Все задачи построения плана действий можно разбить на два типа, которым соответствуют различные модели: планирование в пространстве состояний и планирование в пространства задач.

В первом случае считается заданным некоторое пространство ситуаций. Описание ситуаций включает состояние внешней среды и состояние информационной системы. Ситуации образуют некоторые обобщенные состояния, а действия или изменения во внешней среде приводят к изменению актуализированных в данный момент состояний. Среди обобщенных состояний выделяются начальные состояния (обычно одно) и конечные (целевые) состояния.

Планирование по состояниям. Представление задач в пространстве состояний предполагает задание ряда описаний: состояний, множества операторов и их воздействий на переходы между состояниями, целевых состояний. Описания состояний могут представлять собой строки символов, векторы, двухмерные массивы, деревья,

списки и т. п. Операторы переводят одно состояние в другое. Пространство состояний можно представить как граф, вершины которого помечены состояниями, а дуги - операторами.

Планирование по задачам. Метод планирования по состояниям можно рассматривать как частный случай метода планирования с помощью редукций, ибо каждое применение оператора в пространстве состояний означает сведение исходной задачи к двум более простым, из которых одна является элементарной.

Поиск решения в пространстве задач заключается в последовательном сведении исходной задачи к все более простым до тех пор, пока не будут получены только элементарные задачи. Частично упорядоченная совокупность таких задач составит решение исходной задачи. Декомпозиция задачи на альтернативные множества подзадач удобно представлять в виде И/ИЛИ-графа. В таком графе всякая вершина, кроме конечной, имеет либо конъюнктивно связанные дочерние вершины (И-вершина), либо дизъюнктивно связанные (ИЛИ-вершина). В частном случае, при отсутствии И-вершин, имеет место граф пространства состояний. Конечные вершины являются либо заключительными (им соответствуют элементарные задачи), либо тупиковыми. Начальная вершина (корень И/ИЛИ-графа) представляет исходную задачу. Цель поиска на И/ИЛИ-графе - показать, что начальная вершина разрешима. Разрешимыми являются заключительные вершины (И-вершины), у которых разрешимы все дочерние вершины, и ИЛИ-вершины, у которых разрешима хотя бы одна дочерняя вершина. Разрешающий граф состоит из разрешимых вершин и указывает способ разрешимости начальной вершины. Наличие тупиковых вершин приводит к неразрешимым вершинам. Неразрешимыми являются тупиковые вершины, И-вершины, у которых неразрешима хотя бы одна дочерняя вершина, и ИЛИ-вершины, у которых неразрешима каждая дочерняя вершина.

Разработка модели программного агента

Для реализации технологии взаимодействия программного агента с объектами вычислительной среды в качестве модели программного агента можно использовать фреймовую структуру [4].

Рассмотрим модель программного агента, представленную в виде фрейма. В общем случае такая модель может быть записана в виде:

$$FR < R_1, A_{11}, A_{12,..} A_{1m,} >,... < R_2, A_{21}, A_{22,} ... A_{2m,} >, < R_{k1}, A_{km} > , \qquad (1)$$

где FR – имя фрейма (агента); пара <Ri,Ai> – i -й слот фрейма; Ri, - имя слота, Ai - значение слота.

Структурная схема мультиагентного пространства построенного на основе модели фрейм-программный агент приведена на рис.1.

Слот в модели (1) является компонентой задания программному агента и представлен отношением, состоящем из двух атрибутов <имя>,<значение>. Модифицируем структуру слота и приведем его к такому виду

$$Slot = \langle U, D, dom, r_i, \theta, \Omega \rangle , \qquad (2)$$

где U – множество имен атрибутов, D – множество доменов, *dom* – отображение $U \Rightarrow D$, r_i – модель-кортеж i-го задания агента, Ω – множество операций над отношениями.

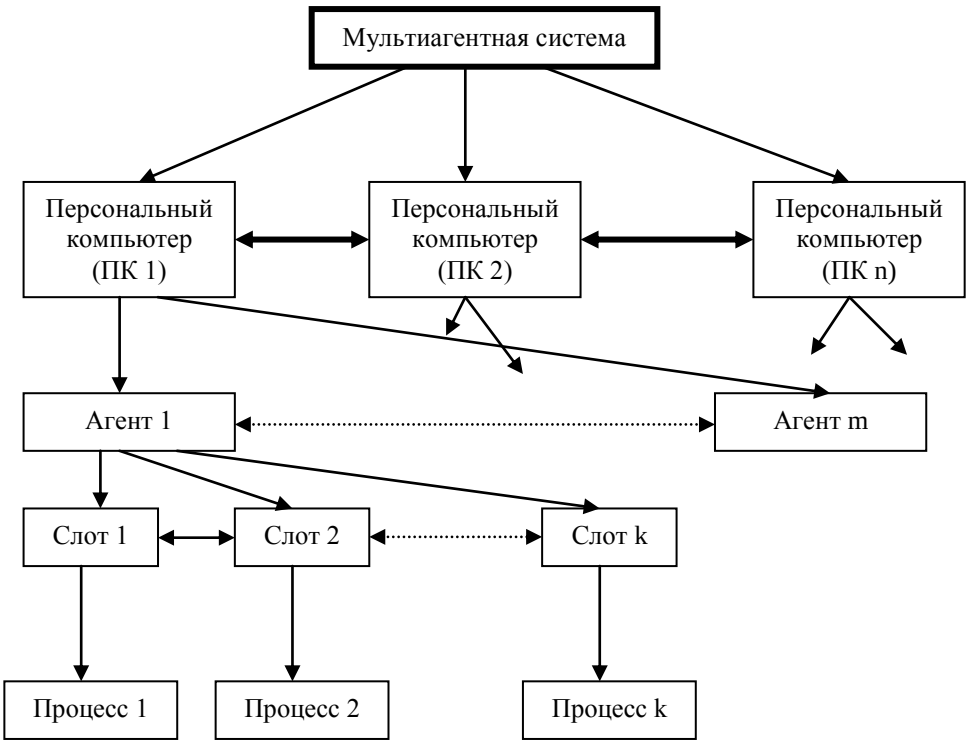

Рис.1. Структурная схема взаимодействия в мультиагентном пространстве

$$r_i = \{\{R\}_i, \Omega_{ij}, V_i\}, \qquad (3)$$

где R_i – множество состояний кортежа r_i, V_i – множество ограничений целостности,

Ω_{ij} – множество операций заданных на R_i, Θ – множество, определяющее начальные условия и признаки выполнения действий в структуре задания.

Так как задания на выполнение определенных функций программным агентом реализуются операциями, определим $W \subseteq D$ – как множество операций композиции, определенных на множестве файлов f_i мультиагентного пространства M, $f_i \in M$, следующим образом $W: M \Rightarrow M$. Множество операций композиции определено на некотором множестве файлов $M = \bigcup_{i=1}^{d} f_i$, при этом файлы могут образовывать как пересекающиеся, так и непересекающиеся подмножества $M^k = \bigcup_{i=1}^{m} f_i$, $M^t = \bigcup_{i=1}^{p} f_i$ ……m, p<=d.

Таким образом, множество M задает подгруппы разных типов файлов, составляющих предметную область мультиагентного пространства администрирования информационной системы. Для файлов входящих в подмножество M^k определим набор допустимых действий, осуществляемых пользователем:

$$d_{ij}^k : \langle w_{ijl}^k \rangle, f_i^{k(n)} \Rightarrow f_i^{k(n+1)}, \qquad (4)$$

где d_{ij}^{k} – тип действия с файлом f_i^k, по преобразованию его из состояния $f_i^{k(n)}$ в состояние $f_i^{k(n+1)}$ набором операций w_{ijl}^{k}. Тип действия также можно представить в виде: $d_{ij}^{k} : \langle w_{ijl}^{k}, f_i^{k(n)}, f_i^{k(n+1)} \rangle$.

Кроме действий и операций по манипулированию файлами внутри подмножества, определим действия и операции между подмножествами, т.е. действия (и операции) преобразования типов.

Действием по преобразованию файла типа f_i^k в файл типа f_i^m, будем называть действие

$$d_{ij}^{k\_m} : \langle s_{ijl}^{k\_m} \rangle, f_i^{k(n)} \Rightarrow f_i^{m(n+1)}, \qquad (5)$$

где $d_{ij}^{k\_m}$ – тип действия с файлом f_i^k, по преобразованию его из типа k состояния $f_i^{k(n)}$ в тип m состояние $f_i^{m(n+1)}$ набором операций $s_{ijl}^{k\_m}$.

На основании модели (2) - (5), может быть реализована модель программного агента в терминах <объекты>, <условия>, <действия>, <приоритет>. Фрейм выступает в виде универсального каркаса или типовой оболочки, в которую могут добавляться функциональные модули-слоты для решения конкретных задач администрирования информационных ресурсов. Каждый слот может формироваться из набора атрибутов базовых типов и выполнять требуемые операции с данными [5].

Пример.

Логическая модель программного агента «Agent» представлена в табл.1 Слот A1 содержит четыре атрибута. Первый атрибут определяет объект действия – файл с именем res.doc. Второй атрибут определяет условие, при котором должно состояться выполнение операции, – если размер файла достигнет 90 Килобайт. Третий атрибут определяет вид операции или вид действия – копировать файл polis.doc на диск D:\ в папку IN. Четвертый атрибут определяет приоритет – важно. Слот A2 выполняет задачу регистрации доступа (время открытия) к файлу базы данных c:\BASE\ST.mdb, протокол доступа хранит в файле d:\Cont\pro.doc. Приоритет – конфиденциально.

Таблица 1

Логическая структура программного агента «Agent»

A1	OBG: c:\MP\res.doc	CON: IF size > 90kb	ACT: COPY to d:\IN	STA: IMP
A2	OBG: c:\BASE\ST.mdb		ACT: PROT d:\Cont\pro.doc	STA: CONF

Выводы

В статье рассмотрены вопросы построения модели программного агента для автономного администрирования информационных ресурсов как абстрактного типа данных. Теоретически обоснован подход к построению такой модели на основании концепции алгебраических систем. Предложенная модель интеллектуального агента модели может использоваться для разработки мультиагентных систем и позволяет

эффективно решать комплекс задач мониторинга и защиты информационных ресурсов вычислительной системы.

ЛИТЕРАТУРА

1. Foundation for Intelligent Physical Agents (FIPA) Spec: DRAFT, version 0.2, Agent Communication language, 1999.

2. Simon A. R. Strategic Database Technology: Management for the Year 2000. - Morgan Kaufmann Publishers, 1995. - P. 100 – 250.

3. Wooldridge M. and N. Jennings (1995) Intelligent agents: theory and practice. The Knowledge Engineering Review. - 10(2). – 1995. – P.115 – 152.

4. Пономаренко Л.А., Філатов В.О. Програмні агентні технології в адмініструванні баз даних // Вісник Київського торговельно-економічного університету. Вип.3. – 2001. - С. 68 – 73.

5. Пономаренко Л.А., Филатов В.А. Динамическое администрирование баз данных с использованием агентных технологий // Научный и произв.-практич. сборник "Труды Одесского политехнического университета". Вып.4 (16). –2001. - С.95 – 97.